



Quality Assurance Automation Standard Best Practices

Recommendations for Development Teams

Revision 1.1
April 28, 2017

Purpose

This document provides recommendations for development teams to leverage during the software development lifecycle to make the best use of Quality Assurance (QA) test automation. When development teams incorporate these recommendations into their software, QA will be able to create automated test cases in a more consistent and reliable manner, which in turn allows for reduced testing time. Using unique identifiers, proper title tags and additional items from the following list promotes quick element identification and provides QA with opportunities to increase automation and reduce QA maintenance of complex object maps. By making the automation process more efficient, defects can be caught and shared with development teams sooner thus improving the efficiency and quality of the overall software development process.

Background

Prior to 2016, DMDC used a commercial off-the-shelf (COTS) application called MicroFocus TestPartner for test case automation. With a push to increase automation coverage in recent years, a decision was made in 2016 to move to an open source solution using SeleniumHQ for browser interaction and a user-friendly interface named RedwoodHQ for test case management. While creating a Java based testing framework, it was observed that there were various development styles incorporated throughout DMDC applications that did not follow a common standard, thus making it more difficult and inefficient to create automated tests for various applications.

Below is a compilation of issues and recommendations that, if adopted, would have a direct impact on the efficiency of automating test cases. While DMDC has implemented the SeleniumHQ and RedwoodHQ solutions, the recommendations presented below are general best practices that will work with other automation solutions.

Recommendations

#	Issue	Recommendation
1	Unique ID attributes in HTML tags	Each ID attribute in an HTML Tag on a page should be unique. Having duplicates makes it more difficult for automation to target specific HTML Elements.

#	Issue	Recommendation
2	Use of Missing or non-unique HTML title tag attributes	The use of uniquely identifying HTML Tag Attributes should be extended whenever possible. Even if an element does not require an ID attribute, it should have some kind of attribute (such as name or ID attribute) that can be used to uniquely identify it when testing. This is especially true when apps use third party frameworks to generate their code. If an ID is being generated, it should have a unique class or name so it can be referred to more easily. This recommendation allows automated testing to be more efficient as automated tests typically use this tag to navigate between pages and windows.
3	Dynamically generated code and unused code	When new sections of a page are generated dynamically, remove them from the page once they are no longer needed. When combined with the problem of having Duplicate ID attribute values, not removing the unused code makes it extremely difficult to make automation code reusable. It should be noted that commented code generally does not interfere with automated testing but keeping code as clean as possible is still a preferred approach.
4	JSP Popups	When developing pop-ups, developers should use JavaScript alerts. JSP's are difficult to test as they require some knowledge of the code and creation of an HttpServletRequest.
5	Compatibility of JavaScript and automation testing tool	When coding JavaScript to react based on field updates, ensure that the JavaScript works with the automation tool for making updates to the field. It is recommended that developers install automation tools used by QA and incorporate them into their toolset. This can be accomplished by executing the test suite for your application from a developer desktop.
6	Compatibility of UI Frameworks and the automation tool	<p>When choosing third party frameworks for generating UI elements, ensure that any element in the generated code can be accessed by Xpath. If an element can't be reached except by accessing it in JavaScript, it slows down automation. For example, Dojo Toolkit with Selenium requires use of custom JavaScript code to be able to trigger the drop down menus.</p> <p>The easiest way of determining whether or not a UI element can be automated is to try and run a click method using the automation tool against it on the developer desktop.</p> <p>It is requested that development teams consider popular frameworks like BootStrap, jQuery, or AngularJS when creating applications to most efficiently leverage the benefits of test automation.</p>

#	Issue	Recommendation
7	Use of HTML Title Tag	DMDC web applications should update their pages to have unique title tags. Automation tools typically use this tag to navigate between pages and windows.
8	Unique ID attributes in HTML tags	Provide names and/or IDs for any element the tester has to interact with to use an application for maintainability. This includes Navigational elements such as buttons, links, form controls, form fields, elements that trigger UI changes (such as elements that expand and contract sections of the page), and elements (such as titles or images) whose existence has to be verified or whose attributes have to be checked. This helps prevent breaking automated test cases when page formatting changes due to error messages, screen flow changes or page re-design. It will also decrease the time needed to adapt to changes by using a simple search and replace versus updating random XPath values.
9	Consistent UI design pattern	To support continuous integration and allow sharing of action collections, applications should be designed to flow in a predictable and similar manner. Suggestion is to include common processes for login, logout and navigation. Current application versions use different names for the same thing and do not always follow a standard.
10	Testability of UI features	<p>If applications contain unique UI features like Captchas, development teams should collaborate with QA early in the development process on options for automation testing for these features.</p> <p>Specific to Captcha, GCE is conducting further research. (Follow this link for a potential solution: http://www.techaspect.com/blog/quality-assurance/automated-captcha-testing-with-selenium-webdriver)</p>

This list will be continuously updated and provided on the Enterprise QA SharePoint site, as new development standard recommendations are identified.